

OpenGL Lecture 1

Your First OpenGL Program

Note: The material covered in this lecture is designated for the final lab exam and will not be included in the theoretical exams.

What is OpenGL?

Open Graphics **L**ibrary is a **cross-language, cross-platform** application programming interface (**API**) for rendering 2D and 3D graphics.

- **Low-Level API:** It provides commands to draw points, lines, and polygons to a frame buffer.
- **State Machine:** OpenGL operates as a state machine; you set the current color, line width, or lighting, and it remains in that state until you change it.

Core Functions of the GL Library

The **GL** library is the foundation of the API, providing the fundamental commands for 3D rendering.

Key Specifications:

- **Purpose:** Manages geometry, appearance, and the states.
- **Scope:** Contains approximately **150 to 250 functions**.
- **Header:** `#include <gl.h>`
- **Syntax:** all functions within this library begin with the prefix `gl` (e.g., `glBegin`, `glColor`).

GL core functions categories

1. Primitive Specification: Used for defining the geometric shapes of a scene.

Example: `glVertex3f(1, 1, 0);` defines a single point in 3D space x,y,z.

2. State Management: Used to set global **attributes** like color or line width that apply to all subsequent drawings.

Example: `glColor3f(1, 0, 0);` sets the "current pen" color to red.

3. Transformations: Handles the mathematical operations required to move or change objects.

Example: `glRotatef(45, 0, 1, 0);` rotates an object 45 degrees around the Y-axis.

4. Lighting: Manages how light sources interact with surface materials for realism.

Example: `glLightfv(GL_LIGHT0, GL_POSITION, light_pos);` sets the location of a specific light source.

Supporting libraries

Standard OpenGL is "window system independent," meaning it doesn't know how to create a window or handle a mouse click. We use helper libraries for that:

Library	Name	Purpose
GLU	OpenGL Utility Library	Used for complex tasks like setting up camera views (<code>gluLookAt()</code>).
GLUT	OpenGL Utility Toolkit	Handles window creation, mouse/keyboard input, and simple 3D shapes (<code>glutCreateWindow()</code>)
GLEW	OpenGL Extension Wrangler	manage the complexities of different hardware and operating systems.

ایں سب کو اگر صحیح سے
ایک جگہ پر

Software and tools

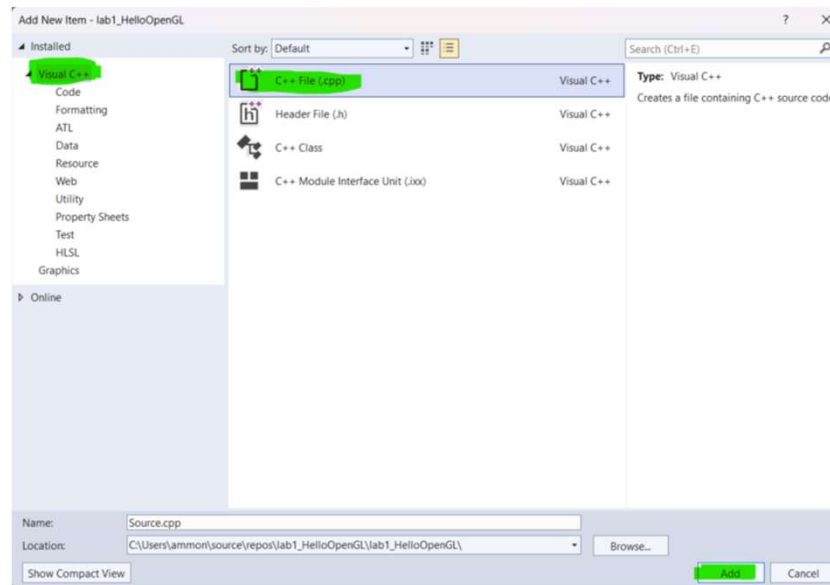
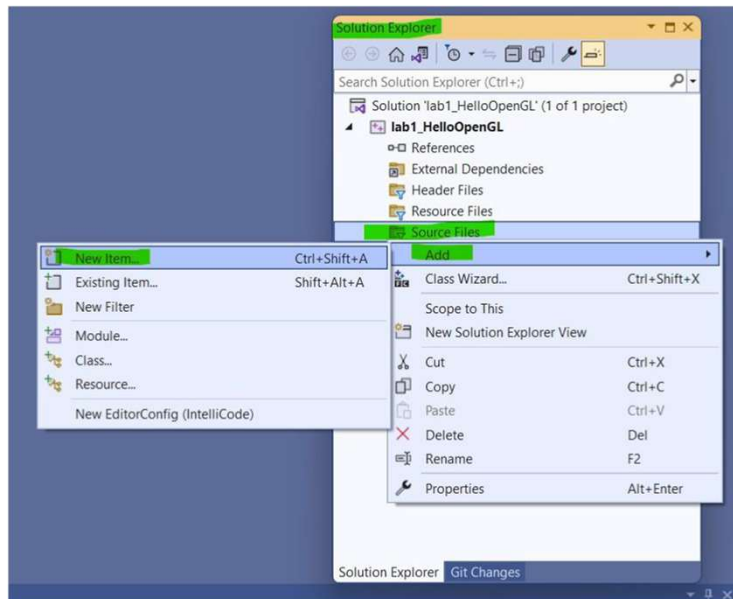
In this course we will use:

- **IDE:** visual Studio
- **Language :** C++
- **Library :** GLUT



Create your First Project

1. Open “Visual Studio”.
2. Create a new project>> C++ >> EmptyProject
3. Choose a name for your project.
4. In the solution explorer right click on Source Files >> add >> New Item
5. Select C++ file (cpp) as shown below.



Write Your First Program

1. Start by including the header files for the main libraries in your program

```
#include <Windows.h>
```

```
#include <GL/glut.h> ← مكتبة بزا المثلث آسن
```

2. Your program must have at least two functions. A drawing function and the main function. Write the headers of these functions as shown below

```
1  #include <Windows.h>
2  #include <glut.h>
3
4  void draw() {           ← how to define
5                          a function
6  }
7
8  int main() {
9
10 }
```

In this course we will use GLUT library (**glut.h**). However, in order to use GLUT we must include **Windows.h**. (order is important)

Important : In your personal device the library header file in another path so you should write :
`#include <glut.h>`

Write Your First Program

3. In the main function body, write the following lines:

```
glutCreateWindow("Hello OpenGL"); // Create a window with the title "Hello OpenGL"
```

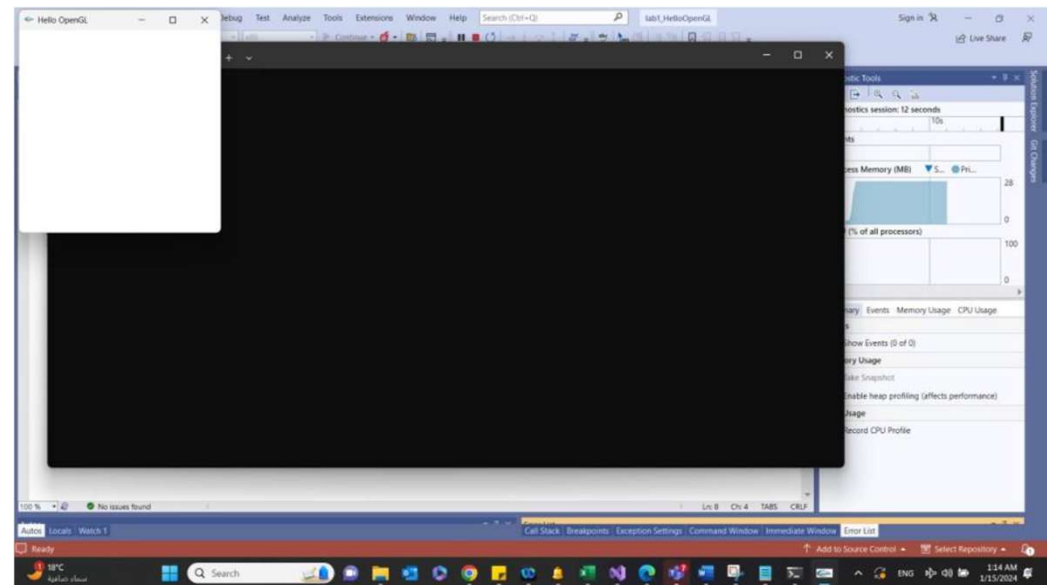
```
glutDisplayFunc(draw); // Used to call the drawing function
```

```
glutMainLoop(); // Enters the GLUT event processing loop. This method keeps the program running  
waiting for any event for example a mouse clicks.
```

has to be the last command in the main

4. Run your program.

Your output should be like the following screen.



Initialization functions

Initialization functions should be called before `glutCreateWindow()`

All these functions start with `glutInit`

Examples:

```
glutInitWindowSize(int width, int height);
```

```
glutInitWindowPosition(int x, int y);
```

```
glutInitDisplayMode(unsigned int mode);
```

Initialization functions:

glutInitDisplayMode(unsigned int mode)

Display modes :

The default values are :

`glutInitDisplayMode (GLUT_RGBA | GLUT_SINGLE)` // use '|' as a separator not ','

Mode type	Constant	description
Colour modes	GLUT_RGB	RGB(red, green, blue) <i>what we will use</i>
	GLUT_RGBA	RGBA(red, green, blue, alpha) > Alpha represents the transparency value.
Memory modes	GLUT_SINGLE	a window with one buffer. always used with <code>glFlush()</code> ; <i>الرسومات العادية</i>
	GLUT_DOUBLE	opens a window with two buffers for animation . used with <code>glSwapBuffers()</code> ; <i>animation</i>
Depth modes	GLUT_DEPTH	opens a window with a depth buffer. used in 3D drawing.

Examples :

for animation we use : `glutInitDisplayMode (GLUT_DOUBLE);`

To use depth buffer: `glutInitDisplayMode (GLUT_DEPTH);`

and default single
writing them in the same line
separating them by or |

Example of main function after Initialization functions:

```
int main() {  
    glutInitWindowSize(400,400);  
    glutInitWindowPosition(200,200);  
    glutInitDisplayMode(GLUT_RGB);  
    glutCreateWindow("Hello OpenGL");  
    glutDisplayFunc(draw);  
    glutMainLoop();  
}
```

Drawing Function

Now let's start drawing, do the following inside **draw()** function:

RESET THE BUFFERS:

To start drawing you **must** reset the buffers first. We will use two functions:

`glClear()`: at the beginning of the function to reset the buffers.

`glFlush()`: the last line in the function. Used to empty the buffer.

Your code should be:

```
void draw() {  
    glClear(GL_COLOR_BUFFER_BIT);  
    /*  
    Your drawing code should be in the middle.  
    */  
    glFlush();  
}
```

Drawing Function :

glClear() accepts one parameter that is the bitwise OR(|) of several values indicating which buffer is to be cleared.

In this course we will cover two buffers:

Buffer name	Description
GL_COLOR_BUFFER_BIT	Indicates the buffers currently enabled for color writing.
GL_DEPTH_BUFFER_BIT	Indicates the depth buffer. (will be covered later)

Examples:

```
glClear(GL_COLOR_BUFFER_BIT);
```

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

Drawing Function :

glClear() accepts one parameter that is the bitwise OR(|) of several values indicating which buffer is to be cleared.

In this course we will cover two buffers:

Buffer name	Description
GL_COLOR_BUFFER_BIT	Indicates the buffers currently enabled for color writing.
GL_DEPTH_BUFFER_BIT	Indicates the depth buffer. (will be covered later)

Examples:

```
glClear(GL_COLOR_BUFFER_BIT);
```

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

Exercise : draw a triangle

by default



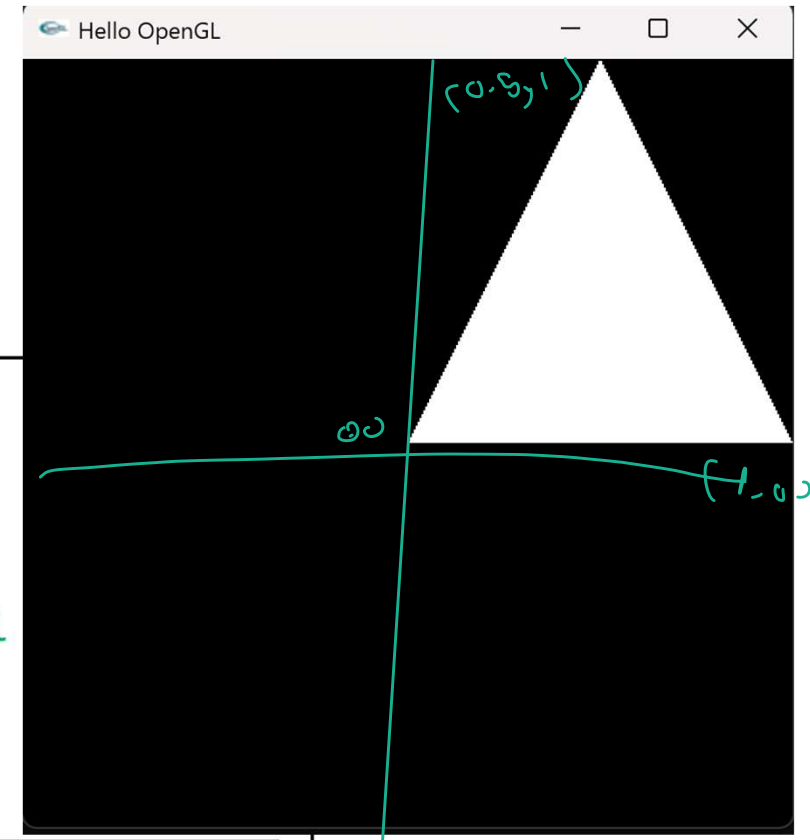
draw = while

Background = black

```
#include <Windows.h>
#include <GL/glut.h>
void draw() {
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_TRIANGLES);
        glVertex2d(0,0);
        glVertex2d(1,0);
        glVertex2d(0.5,1);
    glEnd();
    glFlush();
}
int main() {
    glutInitWindowSize(400,400);
    glutInitWindowPosition(200,200);
    glutInitDisplayMode(GLUT_RGB);
    glutCreateWindow("Hello OpenGL");
    glutDisplayFunc(draw);
    glutMainLoop();
}
```

drawing 2D primitives will
be explained in lab 2

Exercise: try to change the
triangle position in different
places.



Let's add **COLORS**

In OpenGL there are many coloring functions to change the geometry color. These functions start with **glColor** followed by :

- number of parameters: **3** for RGB or **4** for RGBA
- the parameter type : **f** for (0 to 1) and **ub** for (0 to 255)

To make the rectangle red add one of the following functions in any line before glVertex2d(0,0)

`glColor3f(1, 0, 0);`
↓
float 0-1

OR

`glColor3ub(255, 0, 0);`
0-255

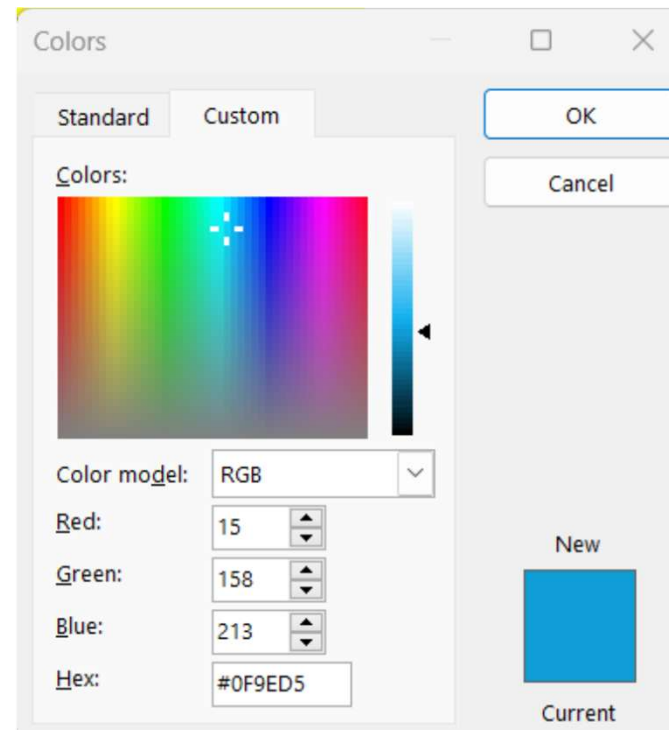
Remember : To convert from **ub** to **f**: $x / 255$, To convert from **f** to **ub**: $x * 255$

Example : `glColor3ub(190, 63, 218) == glColor3f(0.74509, 0.2470, 0.8549)`

How to choose **colors**?

You can use any online RGB tool like <http://www.rgbtool.com>

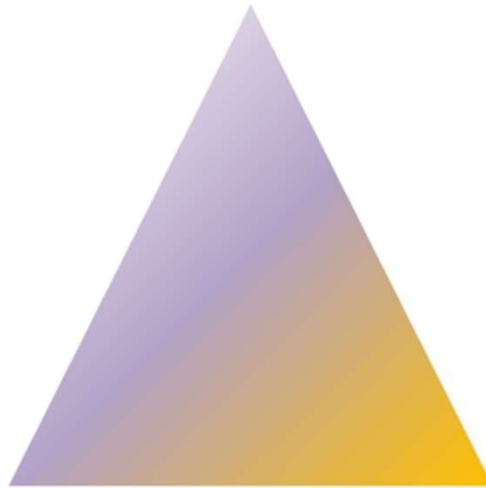
OR use the color tool in any program.



Exercise :

In your previous code:

- Change the color of the previous triangle with any color of your choice.
- Try to draw something like this by choosing a different color before each vertex function.



Background Color using glClearColor:

To change the background color, add `glClearColor()` function in the `main()` function at any line between `glutCreateWindow()` and `glutMainLoop()` or in the `draw()` function before `glClear()`.

```
glClearColor(0.5, 0.5, 0.5, 1); //grey color - uses RGBA system
```

```
// R:0.5 , G:0.5 , B:0.5 , A:1(always 1)
```

```
// it uses only (0 to 1) numbers
```

Homework

Practice the lab content at home with different parameters. For example:

- Change the window size
- Change the window position
- Change the triangle color
- Change the background color.